Comparative Analysis of Some Machine Learning Algorithms for the Classification of Ransomware

Adeniyi, Adedayo Omoniyi[1]., Olabiyisi, Stephen Olatunde[2]., Adepoju, Temilola Morufat[3] and Sanusi, Bashir Adewale[4]

[1, 2, &3] Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Oyo state Nigeria

[4] University of the West of England, Bristol United Kingdom

DOI: https://doi.org/10.51244/IJRSI.2025.120800045

Received: 24 July 2025; Accepted: 30 July 2025; Published: 02 September 2025

ABSTRACT

Ransomware is a serious cybersecurity threat, encrypting data and demanding payment for its release. This study compares six machine learning algorithms, these are Random Forest (RF), Decision Tree (DT), Neural Network (NN), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naive Bayes (NB) for ransomware classification. A GitHub sourced dataset was preprocessed using standard techniques, and feature selection was done using correlation analysis, mutual information, and recursive feature elimination. Models were trained and evaluated using Python's scikit-learn library, assessed on accuracy, precision, recall, F1-score, and ROC-AUC. RF achieved the best performance with 99.98% accuracy and 99.99% ROC-AUC, followed closely by DT and NN. NB performed poorly across most metrics. Results indicate RF as the most effective model for ransomware detection. These findings support the development of intelligent threat detection systems for cybersecurity platforms, cloud infrastructure, and endpoint protection.

Keywords: Comparative Performance, Ransomware, Machine Learning (ML), Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), Feature Selection and Python scikit-learn.

INTRODUCTION

Ransomware attacks have become a significant and escalating threat to information security, targeting systems, data centers, and applications across various sectors. These attacks encrypt critical data and demand ransom payments for its release, often leading to severe operational disruptions, substantial financial losses, and long-term reputational damage to affected organizations (Scaife *et al*., 2016). Traditional ransomware detection methods, such as signature-based approaches, are increasingly inadequate due to the rapid evolution and obfuscation tactics used by modern ransomware variants (Ucci *et al*., 2019). This growing sophistication necessitates the development of more dynamic and intelligent detection mechanisms.

This research aims to address existing limitations in ransomware detection by implementing and comparing these six machine learning algorithms (DT, RF, SVM, KNN, NN, and NB) using a consistent dataset and comprehensive evaluation metrics. A robust feature selection methodology was employed to improve model performance and reduce dimensionality. The goal is to evaluate the classification capabilities of each algorithm using metrics such as Accuracy, Precision, Recall, F1-score, and ROC-AUC, and to provide practical implementation insights for real-world systems.

By incorporating this range of algorithms, the study ensures a comprehensive evaluation of different learning strategies and their effectiveness in the classification of ransomware. The performance of each algorithm is assessed using a unified evaluation framework that includes accuracy, precision, recall, F1-score, and ROC-AUC metrics. This approach supports an informed comparison and contributes to identifying the most suitable techniques for developing effective ransomware detection systems.

In the quest to contribute to knowledge, this study aims to investigate and compare the performance analysis of some machine learning techniques for the classification of ransomware.

The specific objectives are to:

i. Extract relevant ransomware dataset from Github repository using ransomware.csv to facilitate the supervised learning tasks.

ii. Implement and train machine learning models (RF, DT, NN, SVM, KNN and, NB) to classify ransomware data extracted from Github using Python scikit-learn 1.6.1.

iii. Compare the model's performance using Accuracy, Precision, Recall, F1-score, and ROC-AUC.

Related Work

Al-Ruwili *et al.* (2023) explore the impact of machine learning techniques on Android ransomware detection, emphasizing the need for robust classification methods in mobile environments. The authors utilize a combination of static and dynamic features to train their models, including Support Vector Machines and Decision Trees. Their findings reveal that machine learning models can effectively identify ransomware with high accuracy, demonstrating the potential for real-time detection in mobile applications.

Egunjobi *et al.* (2019) conducted a comprehensive study on the classification of ransomware using machine learning algorithms, focusing on the integration of both static and dynamic features to enhance detection accuracy. The authors employed a variety of supervised learning algorithms, including Naïve Bayes, Support Vector Machines, and Random Forest, to evaluate their performance in classifying ransomware samples. The results indicate that SVM and Random Forest achieved the highest accuracy rates of 99.5%, while Naïve Bayes demonstrated an accuracy of 96%. The study highlights the importance of feature selection and the use of confusion matrices to systematically compare the effectiveness of different algorithms, providing valuable insights for future research in ransomware detection.

Khammas *et al.* (2022) conduct a comparative analysis of different machine learning algorithms for ransomware detection, focusing on the effectiveness of feature selection techniques. The study evaluates algorithms such as K-Nearest Neighbors (KNN), Decision Trees, and Support Vector Machines, providing a detailed examination of their performance metrics. The authors find that incorporating advanced feature selection methods significantly improves the accuracy of ransomware detection models.

Masum *et al.* (2022) investigate the classification and detection of ransomware using various machine learning algorithms, including Random Forest, Naïve Bayes, and Neural Networks. The authors emphasize the significance of feature selection and preprocessing in improving model performance. Their results indicate that Random Forest outperforms other algorithms, achieving the highest accuracy in detecting ransomware.

Ngirande *et al.* (2024) present a novel approach to ransomware detection using hybrid machine learning techniques, combining the strengths of various algorithms to enhance classification accuracy. The authors utilize a dataset comprising both benign and malicious samples to train their models, achieving impressive results in detecting ransomware variants. Their findings suggest that hybrid models can outperform traditional single-algorithm approaches, providing a promising direction for future research in ransomware detection.

METHODOLOGY

In the process of evaluating of some selected machine learning models for ransomware classification and to evaluate their performance, the following steps were involved:

Data Collection

The dataset used in this research was collected from a publicly available GitHub repository containing labeled samples of ransomware and benign files. The dataset, named Ransomware.csv, contains various features that are instrumental in distinguishing between benign and ransomware files. The dataset includes relevant features such as file metadata, static attributes, and behavioral indicators. This open-source data ensures access to current and recent ransomware patterns.

Dataset Description

The Ransomware.csv file includes a wide range of features extracted from different types of files. These features are crucial for building machine learning models to classify files as either benign or ransomware. The dataset is structured with the following key attributes:

i. **File Metadata**: Information about the file such as size, type, and creation date.

ii. **Behavioral Features**: Characteristics that describe the behavior of the file, such as the number of read/write operations, network activity, and system modifications.

iii. **Static Features**: Attributes derived from the static analysis of the file, including byte sequences, strings, and header information.

iv. **Dynamic Features**: Data obtained from the dynamic analysis of the file, such as API calls, registry changes, and process activities.

v. **Label**: The target variable indicating whether the file is benign or ransomware.

Data Pre-processing

The raw dataset was cleaned to handle missing values, encode categorical attributes, and standardize numeric features. These pre-processing steps are necessary to ensure data consistency and to make it suitable for training machine learning models. Proper normalization also improves algorithm performance and convergence. The preprocessing steps include:

i. **Handling Missing Values**: Any missing values in the dataset were identified and appropriately handled, either by imputing with mean/median values or by removing the rows/columns with missing data.

ii. **Encoding Categorical Variables**: Categorical variables were encoded into numerical values using techniques such as one-hot encoding or label encoding.

iii. **Feature Standardization**: The features were standardized to have a mean of zero and a standard deviation of one. This step is essential for algorithms that are sensitive to the scale of the data.

Feature Selection

A combination of methods including correlation analysis, mutual information, and recursive feature elimination was used to identify the most relevant features. Redundant and less informative features was removed to enhance accuracy and reduce overfitting. The selected feature set becomes the input for model training.

Various feature selection techniques were applied, including:

i. **Correlation Analysis**: Analyzing the correlation between features and the target variable to identify highly correlated features.

ii. **Mutual Information**: Using mutual information to measure the dependency between features and the target variable.

iii. **Recursive Feature Elimination (RFE)**: Iteratively removing the least important features based on model performance.

Model Initialization

A diverse set of machine learning algorithms were selected to classify the ransomware dataset into benign and ransomware. The models implemented include:

i. **Decision Tree**: The Decision Tree model is initialized using the Gini Index as the splitting criterion, as previously defined in Equation (3.1). This function guides the selection of optimal splits by minimizing impurity at each node of the tree. The DT algorithm splits the dataset into branches based on feature thresholds that maximize information gain or minimize entropy. It creates a hierarchical structure of decisions to classify data points as either ransomware or benign. DTs are initialized using the Gini index or entropy as the splitting criterion and are known for their interpretability and fast execution time.

$$H(S) = -\sum_{i=1}^{c} p_i log^{2(p_i)} \qquad 3.1$$

where $(p_{i)}$ is the proportion of instances belonging to class (i) in set (S). (Masum *et al.*, 2022).

ii. **Random Forest**: RF is an ensemble method that constructs multiple decision trees and aggregates their results for improved accuracy and generalization. RF operates by aggregating multiple decision trees and averaging their predictions, based on the expression provided in Equation (3.2). Each tree is trained on a random subset of the data and features. Model initialization involves setting the number of trees (estimators), the maximum tree depth, and bootstrap sampling. RF reduces overfitting compared to a single decision tree.

$$y = \frac{1}{N}\sum_{i=1}^{N} h_i(x) \qquad 3.2$$

where:

y is the predicted value,

N is the number of trees,

$h_{i(x)}$ is the prediction of the $(i^{\{th\}})$ tree for input (x) (Breiman, 2001).

iii. **Support Vector Machine**: SVM is a powerful classifier that identifies the optimal hyperplane separating the two classes by maximizing the margin between support vectors. It can be adapted for non-linear data using kernel functions such as radial basis function (RBF). The SVM classifier is implemented using the margin maximization formulation outlined in Equation (3.3). The hyperparameters, including the kernel function and regularization constant, are adjusted to optimize separation between the two classes. The SVM model is initialized with parameters such as kernel type, regularization constant (C), and gamma for kernel scaling.

$$min\frac{1}{2}\|w\|^2 \; subject\ to\ y_i(w \cdot x_i + b) \geq 1 \qquad 3.3$$

where:

(w) is the weight vector,

(b) is the bias,

(y_{i}) is the class label for instance (x_{i}) (Ngirande, 2024; Kok *et al.,* 2019).

iv. **K-Nearest Neighbors**: KNN is a distance-based algorithm that classifies a data point based on the majority label of its 'k' nearest neighbors in the feature space. KNN determines the class of a new data point based on the majority vote among its k nearest neighbors. The distance metric guiding this process is previously illustrated in Equation (3.4). The model is initialized by selecting an appropriate value of k and a distance metric, typically Euclidean. KNN is non-parametric and particularly sensitive to feature scaling and noise.

$$\mathrm{d}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad 3.4$$

where (x) and (y) are two data points, and (n) is the number of features (Bawazeer *et al.*, 2021; Abualhaj, 2024). The class of the majority of the (k) nearest neighbors is assigned to the query instance

v. **Neural Network**: NN simulate the behavior of the human brain through layers of interconnected neurons. In this study, a feedforward neural network is implemented with an input layer, one or more hidden layers, and an output layer. The learning process in the feedforward neural network is guided by weight adjustments through backpropagation, as discussed in Equation (3.5). The activation function used at each layer influences how signals are passed forward through the network. The model is initialized with parameters such as activation function (e.g., ReLU), learning rate, optimizer (e.g., Adam), and number of epochs.

$$\mathrm{y} = \mathrm{f}\left(\sum_{i=1}^{n} w_i\, x_i + b\right) \qquad 3.5$$

where:

y is the output,

w_i are the weights,

x_i are the inputs,

b is the bias,

f is the activation function (e.g., sigmoid, ReLU)

Neural Networks are particularly powerful for complex tasks such as image and speech recognition, and they have been successfully applied in malware detection to identify patterns in data that may indicate malicious behavior (Fuyong *et al.,* 2018; Asad *et al.,* 2020).

vi. **Naive Bayes**: NB is a probabilistic classifier based on Bayes' Theorem with an assumption of feature independence. It is particularly efficient for high-dimensional data. Classification is performed using Bayes' Theorem, assuming feature independence, as defined in Equation (3.6). This model calculates posterior probabilities to assign a class label to each instance. The Gaussian Naïve Bayes variant is used when the data is continuous and assumed to follow a normal distribution. The model is initialized without extensive hyperparameter tuning, making it lightweight and fast.

$$P(C|X) = P(X|C) \cdot \frac{P(C)}{P(X)} \qquad 3.6$$

where:

$P(C|X)$ is the posterior probability of class (C) given the features (X)

$P(X|C)$ is the likelihood of the features given class (C)

$P(C)$ is the prior probability of class (C)

$P(X)$ is the total probability of the features. Bold *et al.* (2022) Masum *et al.*, 2022).

Model Training and Classification

The models were trained using the preprocessed dataset and evaluated using various metrics to assess their performance. The steps involved are:

i. **Train-Test Split**: The dataset was split into training and testing sets to evaluate the models' performance on unseen data.

ii. **Model Training**: Each model was trained using the training set.

Performance Evaluation

The performance of the classification models is evaluated using several metrics, all implemented using Scikit-learn. These metrics include Accuracy, Precision, Recall, F1-score, and ROC-AUC. Each metric provides a different perspective on the models' performance, which is crucial for a comprehensive evaluation in the context of ransomware detection. The evaluation metrics employed are:

Accuracy: Accuracy measures the proportion of correctly classified instances (both benign and ransomware) out of the total instances. It is calculated as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (3.7)$$

where:

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

Precision: Precision indicates the proportion of true positive instances among the instances classified as positive (ransomware). It is calculated as:

$$Precision = \frac{TP}{TP+FP} \qquad (3.8)$$

Recall: Recall (or Sensitivity) measures the proportion of true positive instances among all actual positive instances. It is calculated as:

$$Recall = \frac{TP}{TP+FN} \tag{3.9}$$

F1-score: The F1-score is the harmonic mean of Precision and Recall, providing a balanced measure of the models' performance. It is calculated as:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.10}$$

ROC-AUC: The Receiver Operating Characteristic - Area Under Curve (ROC-AUC) measures the ability of the model to distinguish between classes. It is calculated as the area under the ROC curve, which plots the True Positive Rate (Recall) against the False Positive Rate (FPR). The FPR is calculated as:

$$FPR = \frac{FP}{FP + TN} \tag{3.11}$$

RESULT AND DISCUSSION

As shown in table 4.1, the performance analysis of six machine learning models implemented for ransomware detection: Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Neural Network, and Naive Bayes was evaluated using multiple metrics, including Accuracy, Precision, Recall, F1-score, and ROC-AUC.

Table 4.1 Model Performance Evaluation

	Accuracy	Precision	Recall	F1 Score	ROC AUC
Decision Tree	0.99971	0.99952	0.99952	0.99952	0.99966
Random Forest	0.99978	0.99928	1.00000	0.99964	1.00000
SVM	0.99428	0.99008	0.99103	0.99055	0.99976
KNN	0.99377	0.98738	0.99211	0.98974	0.99816
Neural Network	0.99920	0.99952	0.99785	0.99868	0.99996
Naive Bayes	0.47269	0.36453	0.99761	0.53395	0.65940

The analysis of model performance metrics as shown in Figure 4.1 reveals significant variations across different machine learning algorithms. Random Forest emerged as the top performer with exceptional metrics, achieving the highest accuracy of 99.98%, perfect recall (100%), and outstanding precision (99.93%). This superior performance is further validated by its excellent F1-score of 99.96% and near-perfect ROC-AUC value of 99.99%, indicating its robust ability to distinguish between classes.

Decision Tree showed remarkable performance, closely following Random Forest with an accuracy of 99.97%. Its consistent precision and recall values of 99.95% demonstrate balanced prediction capabilities, complemented by strong F1-score and ROC-AUC metrics of 99.95% and 99.97% respectively.

The Neural Network implementation also demonstrated robust performance with 99.92% accuracy, high precision (99.95%), and strong recall (99.78%). Its F1-score of 99.87% and ROC-AUC of 99.99% confirm its effectiveness in classification tasks.

Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) both performed well, with accuracies above 99%. SVM achieved slightly better metrics overall with 99.43% accuracy compared to KNN's 99.38%, though both models demonstrated strong capabilities in classification tasks.

In contrast, Naive Bayes showed significantly lower performance with an accuracy of 47.27% and precision of 36.45%. While it maintained high recall (99.76%), its low precision resulted in a poor F1-score (53.39%) and ROC-AUC (65.94%), indicating substantial limitations in classification accuracy. This underperformance suggests that Naive Bayes' assumptions about feature independence may not hold for this particular dataset.

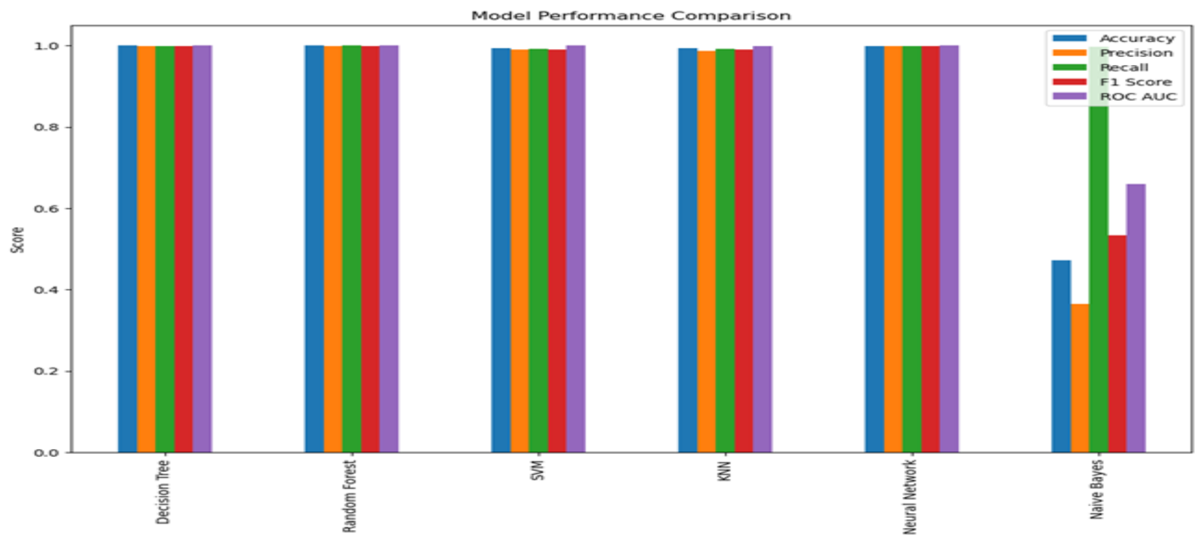

Figure 4.1: Model Performance Comparison

Confusion Matrix Analysis

A detailed analysis of the confusion matrices reveals distinct patterns in classification performance across the different models. The Fig 4.2 shows that Random Forest model demonstrated exceptional performance with zero false negatives out of 27,610 total predictions, achieving perfect recall. This indicates the model never failed to identify positive cases, while maintaining a very low false positive rate with only 11 misclassifications.

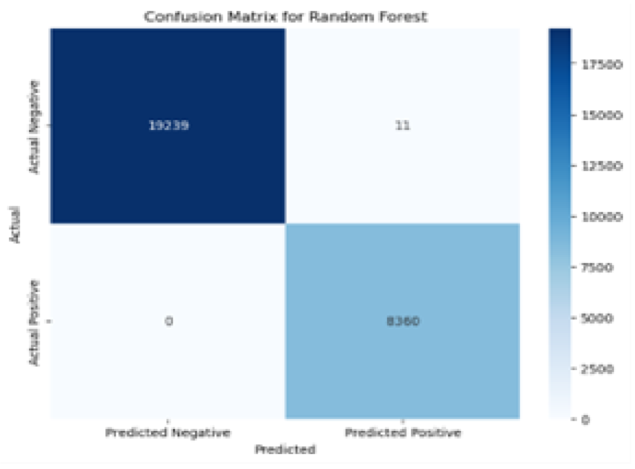

Figure 4.2 Confusion Matrix for RF

The Decision Tree classifier demonstrated a notably balanced performance, recording exactly 4 false positives and 4 false negatives, as illustrated in Figure 4.3. This symmetry in misclassification, along with the model’s high overall accuracy, suggests that it effectively learned the underlying patterns in the data without bias toward either class.

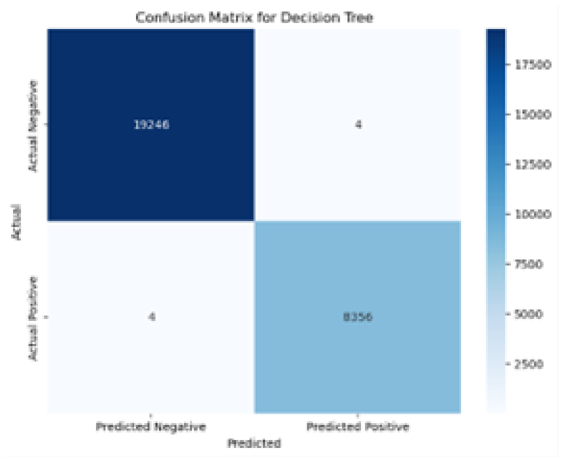

Figure 4.3 Confusion Matrix for DT

As shown in Figure 4.4, the Neural Network's confusion matrix revealed slightly asymmetric errors, with 9 false negatives and 5 false positives. Although these misclassifications remain impressively low, they indicate a slight tendency toward underpredicting the positive class. Nonetheless, the model sustained an excellent overall accuracy exceeding 99.9%.

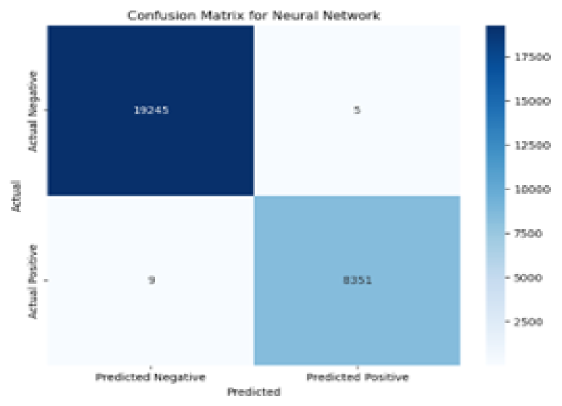

Figure 4.4 Confusion Matrix for NN

The Support Vector Machine (SVM) exhibited higher, yet balanced, misclassification rates with 83 false positives and 75 false negatives as depicted in Figure 4.5. This near-symmetrical distribution of errors suggests that, despite a greater number of misclassifications compared to the top-performing models, the SVM maintained a good balance between the two classes.

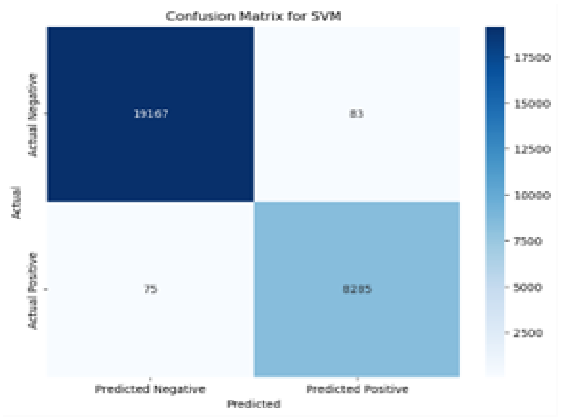

Figure 4.5 Confusion Matrix for SVM

The K-Nearest Neighbors (KNN) algorithm showed a notable tendency toward false positives, with 106 instances compared to 66 false negatives, as shown in Figure 4.6. This suggests a slight bias toward positive predictions, though the overall error rates remain low in the context of the dataset size. The higher number of misclassifications compared to other models (except Naive Bayes) aligns with its slightly lower accuracy metrics.

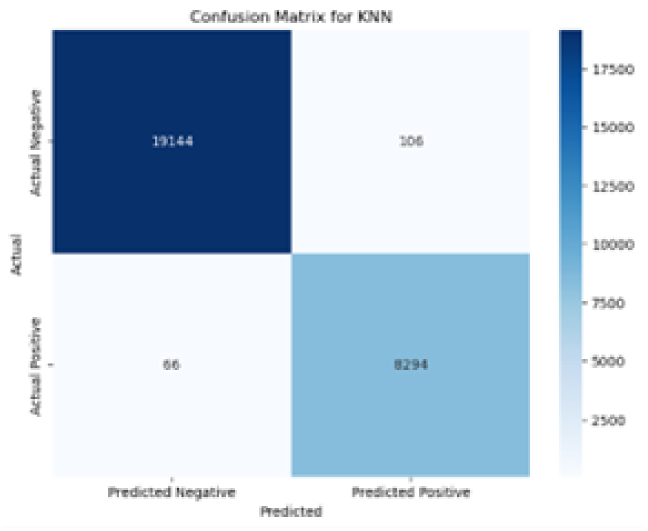

Figure 4.6 Confusion Matrix for KNN

The Naive Bayes model's confusion matrix, as shown in Figure 4.7, reveals significant challenges in classification performance. With 11,866 false positives, the model showed a strong tendency to overpredict positive cases, leading to poor precision. While it correctly identified most positive cases (high recall) with only 20 false negatives, this came at the cost of misclassifying a large number of negative cases as positive. This imbalance in prediction errors explains the model's low accuracy of 47.27% and suggests that the Naive Bayes assumptions about feature independence were not appropriate for this classification task. The high number of false positives indicates that the model struggled to properly differentiate between classes, making it the least reliable among all tested models for this particular application.

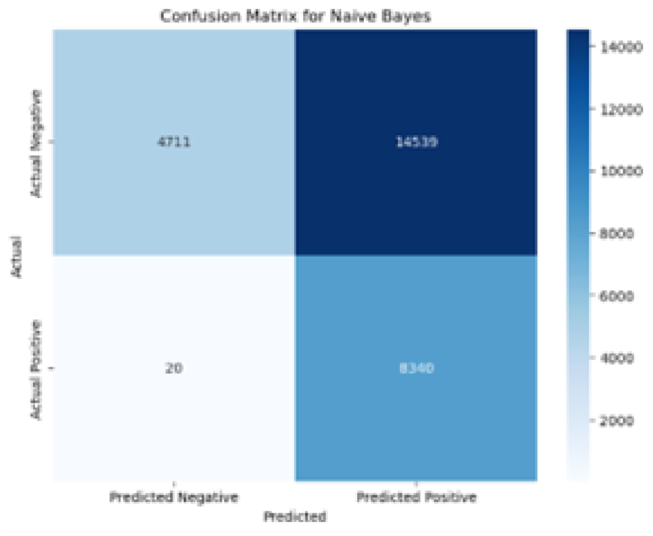

Figure 4.7 Confusion Matrix for NB

Feature Importance Analysis

The analysis of feature importance across all six machine learning models, as illustrated in Figure 4.8, reveals significant insights into the key determinants of ransomware detection. The results demonstrate varying degrees of feature significance across different models, with some features consistently emerging as crucial predictors.

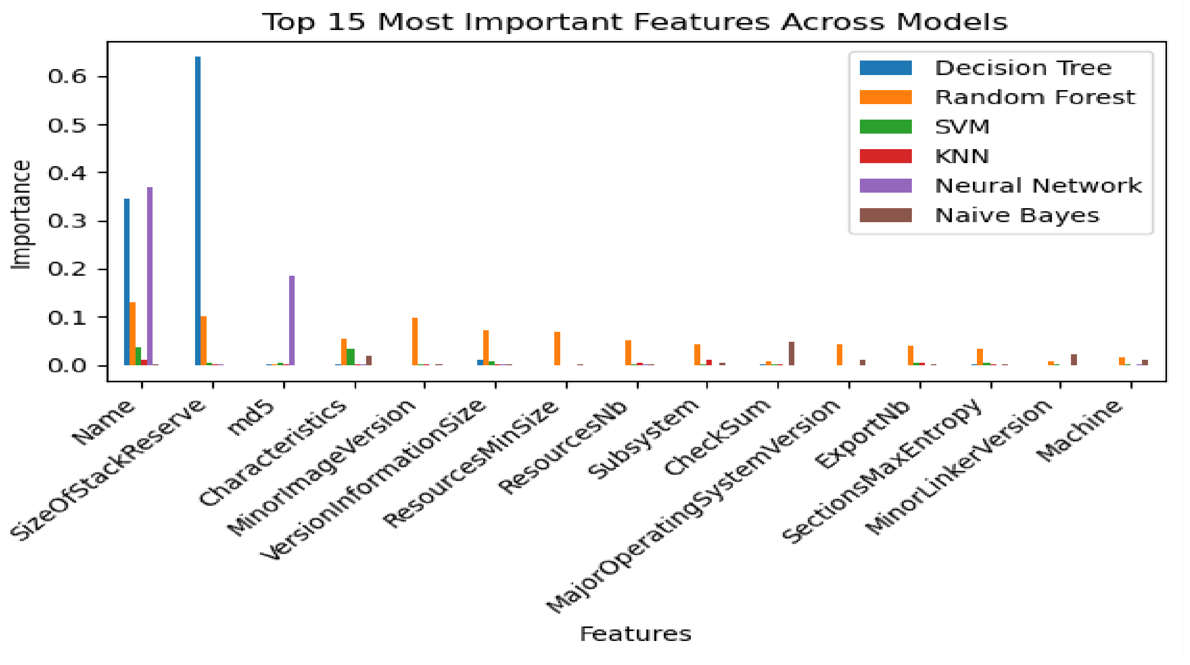

Figure 4.8 Feature Importance Analysis

The Decision Tree model heavily relied on two dominant features: Size Of Stack Reserve (63.98%) and Name (34.64%), which together accounted for approximately 98% of the model's decision-making process. This strong concentration on just two features suggests that the Decision Tree model found these characteristics particularly discriminative for ransomware classification.

The Random Forest model exhibited a more distributed feature importance pattern, utilizing a broader range of features in its decision-making process. The top contributing features included Name (13.02%), Size Of Stack Reserve (10.16%), and Minor Image Version (9.92%). Additionally, features such as Version Information Size (7.02%), Resources Min Size (6.89%), and Characteristics (5.51%) showed moderate importance, demonstrating the model's ability to leverage multiple file characteristics for classification.

The Neural Network model showed a distinct preference for specific features, with Name having the highest importance (36.90%), followed by md5 (18.46%). This concentration on these particular features suggests that the neural network identified strong patterns in these characteristics for distinguishing between benign and malicious files. The Support Vector Machine (SVM) demonstrated more balanced feature importance distributions, with Name (3.64%) and Characteristics (3.28%) being the most influential features. This more evenly distributed importance suggests that SVM relies on a broader set of features for its classification decisions, although with lower individual feature weights compared to other models.

The K-Nearest Neighbors (KNN) algorithm showed relatively low feature importance values across all features, with Subsystem (0.98%) and Name (1.12%) being the most significant. This pattern suggests that KNN's classification decisions are based on multiple features with similar levels of importance, rather than being dominated by specific characteristics. Naive Bayes showed the most distinct feature importance pattern, with Check Sum (4.86%) being its most influential feature, followed by Minor Linker Version (2.34%) and Characteristics (1.86%). However, many features showed very low or negative importance values, indicating that the model might not effectively utilize the full range of available features, which could explain its lower overall performance.

A notable observation as seen in the Table 4.2 across all models is the consistent importance of the Name feature, appearing as a top contributor in most models except Naive Bayes. This suggests that file naming patterns carry significant information for ransomware detection. Similarly, structural characteristics like Size Of Stack Reserve and various version information features proved important across multiple models, indicating their reliability as indicators of malicious software.

The analysis also reveals that certain features, such as Loader Flags, Number Of Rva And Sizes, and Section Alignment, consistently showed minimal importance across all models, suggesting they might be less relevant for ransomware detection. This insight could be valuable for feature selection in future model optimizations.

Table 4.2: Feature Importance of first 10 features across various models

	DCM	RFM	SVM	KNN	NNM	NBM
Name	0.3464	0.1302	0.0364	0.0112	0.3690	0.0006
Size Of Stack Reserve	0.6398	0.1016	0.0056	0.0004	0.0004	-0.0002
md5	0.0001	0.0014	0.0056	0.0024	0.1846	-0.0004
Characteristics	0.0013	0.0551	0.0328	0.0014	0.0024	0.0186
Minor Image Version	0.0000	0.0992	0.0012	0.0024	0.0000	0.0006
Version Information Size	0.0096	0.0702	0.0062	0.0024	0.0002	0.0022
Resources Min Size	0.0000	0.0689	0.0000	0.0000	0.0000	0.0014
Resources Nb	0.0000	0.0525	0.0012	0.0040	0.0008	0.0010
Subsystem	0.0000	0.0414	0.0018	0.0098	0.0000	0.0056
Check Sum	0.0017	0.0071	0.0002	0.0004	0.0000	0.0486

Discussion of Results

The current research findings indicate that Random Forest outperformed all other machine learning algorithms tested, achieving an accuracy of 99.98%, perfect recall, and an F1-score of 99.96%. This aligns with the results presented by (Egunjobi *et al.*, 2019), who also found Random Forest to be highly effective in classifying ransomware, achieving accuracy rates of 99.5% (Egunjobi *et al.*, 2019). The consistency in performance across different studies highlights the robustness of Random Forest as a preferred algorithm for ransomware detection. In contrast, Naïve Bayes exhibited significantly lower performance in the current research, with an accuracy of only 47.27%. This underperformance echoes findings from various studies, including those by (Masum *et al.*, 2022), where Naïve Bayes was noted for its limitations in effectively classifying ransomware due to its assumptions about feature independence (Masum *et al.*, 2022).

The Decision Tree model in the current study demonstrated remarkable performance, closely following Random Forest with an accuracy of 99.97%. This is consistent with the findings of Khammas *et al.* (2022), who emphasized the effectiveness of Decision Trees in ransomware detection (Sharma *et al.*, 2021). The balanced performance metrics of the Decision Tree, including its low false positive and false negative rates, suggest that it can effectively learn the underlying patterns in the data, a characteristic also noted in previous research. The current study's findings regarding the Neural Network's performance, achieving an accuracy of 99.92%, further support the notion that deep learning models can be effective in ransomware classification, as highlighted by (Sharma *et al.*, 2021), who noted the advantages of deep learning over traditional machine learning methods (Sharma *et al.*, 2021).

The comparative analysis of model performance metrics reveals that while SVM and KNN also performed well, their accuracies were slightly lower than those of Random Forest and Decision Tree. This observation is consistent with the work of (Aurangzeb *et al.*, 2021), which indicated that SVM can achieve promising results but may not always outperform ensemble methods like Random Forest (Aurangzeb *et al.*, 2021). The current research's findings on the confusion matrices provide a deeper understanding of the models' classification capabilities, particularly the high number of false positives associated with Naïve Bayes, which aligns with the challenges noted in previous studies regarding the algorithm's reliability in distinguishing between classes (Masum *et al.*, 2022).

Furthermore, the current research highlights the importance of feature selection, with Random Forest utilizing a broader range of features compared to the Decision Tree, which relied heavily on only two features. This observation resonates with the findings of (Ngirande, 2024), who emphasized the significance of comprehensive feature analysis in enhancing model performance (Ngirande, 2024). The consistent importance of the "Name" feature across multiple models in the current study suggests that certain characteristics are critical indicators for ransomware detection, reinforcing the conclusions drawn by other researchers regarding the relevance of specific features in classification tasks (Sharma *et al.*, 2021; Masum *et al.*, 2022).

In summary, the current research findings not only corroborate previous studies regarding the effectiveness of various machine learning algorithms in ransomware detection but also highlight existing gaps in the literature, such as limited algorithm diversity and insufficient comparative analyses. Addressing these gaps will be essential for advancing the field and developing more robust detection systems capable of adapting to the evolving landscape of ransomware threats.

CONCLUSION

This research aims to evaluate and compare the performance of machine learning models namely; Random Forest (RF), Decision Tree (DT), Neural Network (NN), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naïve Bayes (NB) for the classification of ransomware data obtained from a GitHub repository. The models are implemented using Python's Scikit-learn library, and their performance is assessed using evaluation metrics such as Accuracy, Precision, Recall, F1-score, and ROC-AUC. The results demonstrated that the Random Forest emerged as the top performer with exceptional metrics: 99.98% accuracy, 100% recall, and 99.93% precision, demonstrating perfect detection of positive cases with only 11 misclassifications out of 27,610 predictions. Decision Tree showed remarkable consistency with 99.97% accuracy and balanced precision and recall values of 99.95%, exhibiting perfect symmetry in misclassification with exactly 4 false positives and 4 false negatives. Neural Network achieved 99.92% accuracy with high precision (99.95%) and recall (99.78%), showing slight asymmetry in error distribution.

SVM and KNN demonstrated strong performance with accuracies above 99%, with SVM achieving 99.43% accuracy and KNN 99.38%. Naive Bayes significantly underperformed with 47.27% accuracy and 36.45% precision, showing a strong bias toward false positives (11,866 cases).

This research has demonstrated the superior effectiveness of ensemble and tree-based methods in ransomware detection. The exceptional performance of Random Forest and Decision Tree algorithms, combined with strong results from Neural Networks and SVM, establishes a robust framework for automated ransomware detection. The study's comprehensive comparison of multiple algorithms provides valuable insights into their relative strengths and limitations in cybersecurity applications.

Based on the findings of this study, the following recommendations are proposed for future research and practical applications:

i. Explore advanced ensemble methods and deep learning architectures to potentially improve upon the current performance benchmarks.

ii. Investigate feature engineering techniques to enhance model performance, particularly focusing on the most influential features identified across multiple algorithms.

iii. Develop real-time detection systems leveraging the high-performing models, with particular emphasis on Random Forest and Neural Network implementations.

iv. Expand the dataset to include emerging ransomware variants and conduct regular model retraining to maintain effectiveness against evolving threats.

v. Foster collaboration between machine learning experts and cybersecurity professionals to translate these findings into practical defense mechanisms.

REFERENCES

1. Abualhaj, M. M., Abu-Shareha, A. A., Shambour, Q. Y., Al-Khatib, S. N., and Hiari, M. O. (2024). Tuning the k value in k-nearest neighbors for malware detection. IAES International Journal of Artificial Intelligence (IJ-AI), 13(2), 2275–2282. https://doi.org/10.11591/ijai.v13.i2.pp2275-2282
2. Al-Ruwili, A. S. M., & Mostafa, A. M. (2023). Analysis of Ransomware Impact on Android Systems using Machine Learning Techniques. International Journal of Advanced Computer Science and Applications, 14(11), 775–785. https://doi.org/10.14569/IJACSA.2023.0141178
3. Asad, A. B., Mansur, R., Zawad, S., Evan, N., and Hossain, M. I. (2020). Analysis of malware prediction based on infection rate using machine learning techniques. 2020 IEEE Region 10 Symposium (TENSYMP). https://doi.org/10.1109/TENSYMP50017.2020.9230624
4. Aurangzeb, S., Rais, R. N. B., Aleem, M., Islam, M. A., and Iqbal, M. A. (2021). On the classification of Microsoft-Windows ransomware using hardware profile. *PeerJ Computer Science*, *7*, e361. https://doi.org/10.7717/peerj-cs.361
5. Bawazeer, O., Helmy, T., and Al-Hadhrami, S. (2021). Malware detection using machine learning algorithms based on hardware performance counters: Analysis and simulation. Journal of Physics: Conference Series, 1962(1), 012010. https://doi.org/10.1088/1742-6596/1962/1/012010
6. Bold, R., Al-Khateeb, H., and Ersotelos, N. (2022). Reducing false negatives in ransomware detection: A critical evaluation of machine learning algorithms. Applied Sciences, 12(24), 12941. https://doi.org/10.3390/app122412941
7. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324
8. Egunjobi, S., Parkinson, S., and Crampton, A. (2019). Classifying ransomware using machine learning algorithms. In *Intelligent Data Engineering and Automated Learning – IDEAL 2019* (pp. 45–52). Springer. https://doi.org/10.1007/978-3-030-33617-2_5
9. Fuyong Xing, Yuanpu Xie, Hai Su, Fujun Liu, Lin Yang (2018). "Deep Learning in Microscopy Image Analysis: A Survey." IEEE Transactions on Neural Networks and Learning Systems, 29(10), 4550–4568. https://doi.org/10.1109/TNNLS.2017.2766168
10. Khammas, B. M. (2022). Comparative analysis of various machine learning algorithms for ransomware detection. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *20*(1), 43–52. https://doi.org/10.12928/telkomnika.v20i1.18812
11. Kok, S., Abdullah, A., Jhanjhi, N. Z., and Supramaniam, M. (2019). Prevention of crypto-ransomware using a pre-encryption detection algorithm. Computers, 8(4), 79. https://doi.org/10.3390/computers8040079
12. Masum, M., Faruk, M. J. H., Shahriar, H., Qian, K., Lo, D., and Adnan, M. I. (2022). Ransomware classification and detection with machine learning algorithms. *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 0316–0322. https://doi.org/10.1109/CCWC54503.2022.9720869
13. Ngirande, H., Muduva, M., Chiwariro, R., and Makate, A. (2024). Detection and analysis of Android ransomware using the support vector machines. *International Journal for Research in Applied Science and Engineering Technology*, *12*(1), 241–252. https://doi.org/10.22214/ijraset.2024.57885
14. Scaife, N., Carter, H., Traynor, P., & Butler, K. R. B. (2016). Cryptolock (and Drop It): Stopping Ransomware Attacks on User Data. 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS).
15. Sharma, S., Kumar, R., and Krishna, C. R. (2021). A survey on analysis and detection of Android ransomware. *Concurrency and Computation: Practice and Experience*, *33*(16), e6272. https://doi.org/10.1002/cpe.6272
16. Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. Computers & Security, 81, 123–147.